An Ambient Agent Model for Automated Mindreading by Identifying and Monitoring Representation Relations

Alexei Sharpanskykh and Jan Treur

Vrije Universiteit Amsterdam, Department of Artificial Intelligence, Agent Systems Research Group
De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands
Email: {sharp, treur}@cs.vu.nl URL: http://www.cs.vu.nl/~{sharp, treur}

ABSTRACT
In this paper an ambient agent model is presented for automated mindreading based on monitoring a human's interaction with his or her environment. Within this agent model, a cognitive model for the human is assumed to be available. Monitoring foci on a human's interaction with the environment are determined from this cognitive model by automatically deriving representation relations for cognitive states expressed by temporal predicate logical specifications. From these temporal expressions the events are derived that are to be monitored, and from the monitoring information on these events the representation expressions are verified automatically.

Categories and Subject Descriptors
H.1.2 User/Machine Systems, Human factors. I.2.11 Distributed Artificial Intelligence, Intelligent agents.

General Terms
Design, Human Factors, Verification.

Keywords
Theory of Mind, mindreading, representation relation, monitoring, verification, ambient agent model.

1. INTRODUCTION
Applications within the area of Ambient Intelligence address technology to contribute to personal care for safety, health, performance, and wellbeing; e.g., [1], [2], [17]. Such applications make use of possibilities to acquire sensor information about humans and their functioning, and knowledge for analysis of such information. Based on this, ambient devices can respond by undertaking actions in a knowledgeable manner that improve the human's, safety, health, performance, and wellbeing. Two of the crucial aspects of such applications which are addressed in this paper are the decisions on what to be monitored (monitoring foci), and how to derive conclusions about the human's states from such acquired monitoring information. In particular, when an ambient system is to determine the human's cognitive state, it is performing mindreading.

PETRA'08, July 15-19, 2008, Athens, Greece.

In this paper a component-based ambient agent model is presented for automated mindreading. Within this model agent monitoring foci are determined from a cognitive model by automatically deriving representation relations for the human's cognitive states. Within Philosophy of Mind a representation relation relates the occurrence of an internal cognitive state property of a human at some time point to the occurrence of other (internal and/or externally observable) state properties at the same or at different timepoints. In the ambient agent model, these representation relations are expressed as temporal predicate logical specifications. From these temporal expressions the externally observable events are derived that are to be monitored: events that are relevant to the human's generation of the cognitive states addressed. From the monitoring information on these events the ambient agent verifies the representation expressions, and thus concludes whether or not the human is in such a state. The proposed approach allows to identify cognitive states of the human at any time point. Furthermore, in case an internal state has been identified that may affect the functioning and/or wellbeing of the human in a negative way, appropriate actions may be undertaken by the ambient agent.

The ambient agent model for mindreading has been designed as a specialisation of a more general component-based ambient agent model for human-like ambience (cf. [5]), which is based on component-based agent design principles as presented in [10]. Within this agent model, an explicitly represented cognitive model of the human's functioning is assumed, expressed in the form of causal and dynamical relationships between cognitive states and behavioural aspects (i.e., specific forms of interaction with the environment by sensing and acting). The design has been specified in the form of an executable component-based agent-based model that can be (and has been) used for simulation and prototyping.

The paper is organised as follows. First, the modelling approach is introduced in Section 2. In Section 3 the monitoring component is described with subcomponents for monitoring foci determination and monitoring foci verification. Section 4 describes in more detail how exactly the monitoring foci are obtained, by deriving from the given cognitive model representation relations for cognitive states. Section 5 illustrates this by two more extensive examples: one concerning states of core consciousness [11], and one concerning a learning state. In Section 6 simulation results using the model are described, for the case of a learning state. Finally, Section 7 is a discussion.

2. MODELLING APPROACH

This section briefly introduces the modelling approach used. To specify the model conceptually and formally, the agent-oriented perspective is a suitable choice.

Component-Based Ambient Agent Model

An ambient agent is assumed to maintain knowledge about certain aspects of human functioning, and information about the current state and history of the world and other agents. Based on this knowledge it is able to have some understanding of the human processes, and can behave accordingly. Based on the component-based Generic Agent Model (GAM) presented in [10], a model for ambient agents (AAM) was designed [5]. Within AAM, as in GAM the component *World Interaction Management* takes care of interaction with the world, the component *Agent Interaction Management* takes care of communication with other agents. Moreover, the component *Maintenance of World Information* maintains information about the world, and the component *Maintenance of Agent Information* maintains information about other (for example, human) agents. In the component *Agent Specific Task*, specific tasks can be modelled. Adopting this component-based agent model GAM, the ambient agent model has been obtained as a refinement in the following manner.

The component *Maintenance of Agent Information* has three subcomponents in AAM. The subcomponent *Maintenance of a Dynamic Agent Model* maintains a cognitive model represented by causal and temporal relationships for the human's functioning. Note that in this way within the agent model that specifies the ambient software/hardware agent, another model is included which describes the human; this is a form of recursive modelling. The subcomponent *Maintenance of an Agent State Model* maintains a snapshot of the (current) state of the human. As an example, this may model the human's gaze focussing state, a belief, or an intention. The subcomponent *Maintenance of an Agent History Model* maintains the history of the (current) state of the human. This may for instance model intentions over time.

Similarly, the component *Maintenance of World Information* has three subcomponents for a dynamic world model, a world state model, and a world history model, respectively. Moreover, the component *Agent Specific Task* has the following three subcomponents: *Simulation Execution* extends the information in the human state model based on the internally represented dynamic cognitive model for the human's functioning, *Process Analysis* assesses the current state of the agent, and *Plan Determination* determines whether action has to be undertaken, and, if so, which ones. Finally, as in the model GAM, the components *World Interaction Management* and *Agent Interaction Management* prepare (based on internally generated information) and receive (and internally forward) interaction with the world and other agents (including the human).

State Ontologies and Temporal Relations

To express the information involved in the agent's internal processes, the ontology shown in Table 1 was specified. *An ontology* is a signature specified by a tuple $<S_1,\ldots, S_n,\ldots, C, f, P, arity>$, where S_i is a sort for $i=1,\ldots, n$, C is a finite set of constant symbols, f is a finite set of function symbols, P is a finite set of predicate symbols, arity is a mapping of function or predicate symbols to a natural number. Furthermore, for each component of an agent input, output and internal ontologies are defined. Information transmitted between input and output interfaces of components may be mapped automatically using information links connecting the components. Furthermore, information obtained at the input of a component may be mapped automatically to information generated at the component's output. In particular, the input ontology of the World Interaction Management component contains a predicate to specify the results of (passive and active) observation from the world, and its output ontology contains predicates to specify actions and active observation performed in the world (see Table 1). To store the results of observation, the World Interaction Management component maps observation_result(I, S) at its input to new_world_info(I', S') at its output, which is provided to the Maintenance of World Information component via a link. This link performs the mapping of new_world_info(I, S) to belief(I', S'), which is a predicate that belongs to the input, output and internal ontologies of the Maintenance of World Information component. Stored beliefs may be further provided to other components of the agent (e.g., to the Agent Specific Task component). Similarly, the component Agent Interaction Management receives at its input information communicated from other agents and may initiate a communication at its output using the corresponding communication predicates from Table 1. The communicated information is mapped from communicated_by(I, S, A) to new_agent_info(I', S') at the output of the component Agent Interaction Management. This information may be stored by transmitting it to the Maintenance of Agent Information component by a link. This link describes the mapping of new_agent_info(I, S) to belief(I', S'), which is a predicate that belongs to the input, output and internal ontology of the Maintenance of Agent Information component.

The subcomponent Maintenance of a Dynamic Agent Model contains cognitive models that are specified as sets of beliefs using a part of the ontology from Table 1. As an example

belief(leads_to_after(I:INFO_ELEMENT, J:INFO_ELEMENT, D:REAL), pos)

is an expression based on this ontology which represents that the agent has the knowledge that state property I leads to state property J with a certain time delay specified by D.

Table 1. A part of the ontology used within the Ambient Agent Model

Ontology element	**Description**
belief(I:INFO_ELEMENT, S:SIGN)	information I with sign S is believed
to_be_observed(I:INFO_ELEMENT)	I is to be observed in the world (active observation)
observation_result(I:INFO_ ELEMENT, S: SIGN)	I with sign S is observed in the world
communicated_by(I:INFO_ ELEMENT, S:SIGN, A:AGENT)	I with sign S is communicated by A
to_be_communicated_to(I:INFO_ ELEMENT, S:SIGN, A:AGENT)	I with sign S is communicated to A
new_world_info(I: WORLD_INFO_ ELEMENT, S:SIGN)	new information about the world I with sign S
new_agent_info(I: AGENT_INFO_ ELEMENT, S:SIGN)	new information I about an agent with sign S
to_be_performed(A:ACTION)	action A is to be performed
leads_to_after(I:INFO_ELEMENT, J:INFO_ELEMENT, D:REAL)	state property I leads to state property J after D
at(I:INFO_ELEMENT, T:TIME)	property I holds at time T

Furthermore, cognitive specifications may contain temporal relations. The modelling approach to model temporal expressions within the agent is based on the Temporal Trace Language (TTL) for formal specification and verification of dynamic properties [8], [18]. This reified temporal predicate logical language supports formal specification and analysis of dynamic properties, covering both qualitative and quantitative aspects. The agent's dynamics is represented in TTL as an evolution of states of the agent over time. A state of the agent (or of its component) is characterized by a set of state properties expressed over (state) ontology Ont that hold. In TTL state properties are used as terms (denoting objects). To this end the state language is imported in TTL as follows: For every sort S from the state language the following sorts are introduced in TTL: the sort S^{VARS}, which contains all variable names of sort S, the sort S^{GTERMS}, which contains names of all ground terms constructed using sort S; sorts S^{GTERMS} and S^{VARS} are subsorts of sort S^{TERMS}. Sort STATPROP contains names for all state formulae. The set of function symbols of TTL includes $\wedge, \vee, \rightarrow, \leftrightarrow$: STATPROP x STATPROP $\rightarrow$ STATPROP; not: STATPROP $\rightarrow$ STATPROP, and $\forall, \exists$: S^{VARS} x STATPROP $\rightarrow$ STATPROP, of which the counterparts in the state language are Boolean propositional connectives and quantifiers. Further we shall use $\wedge, \vee, \rightarrow, \leftrightarrow$ in infix notation and $\forall, \exists$ in prefix notation for better readability. To represent dynamics of a system sort TIME (a set of time points) and the ordering relation > : TIME x TIME are introduced in TTL. To indicate that some state property holds at some time point the relation at: STATPROP x TIME is introduced. The terms of TTL are constructed by induction in a standard way from variables, constants and function symbols typed with all before-mentioned sorts. The set of *atomic TTL-formulae* is defined as:

(1) If t is a term of sort TIME, and p is a term of the sort STATPROP, then at(p, t) is an atomic TTL formula.
(2) If τ_1, τ_2 are terms of any TTL sort, then $\tau_1 = \tau_2$ is an TTL-atom.
(3) If t_1, t_2 are terms of sort TIME, then $t_1 > t_2$ is an TTL-atom.

The set of well-formed TTL formulae is defined inductively in a standard way using Boolean connectives and quantifiers over variables of TTL sorts. The language TTL has the semantics of many-sorted predicate logic [14]. A special software environment has been developed for TTL, featuring a Property Editor for building TTL properties and a Checking Tool that enables automated formal verification of such properties against a set of traces.

To specify executable models (e.g., models that can be used for simulation), a sublanguage of TTL called LEADSTO [7] has been developed. This language enables modelling direct temporal dependencies between two state properties in successive states in the format:

$$\alpha \twoheadrightarrow_{e,\, f,\, g,\, h} \beta$$

here α and β are state properties in form of a conjunction of atoms or negations of atoms, and e, f, g, h non-negative real numbers. This format is interpreted as follows: if state property α holds for a certain time interval with duration g, then after some delay (between e and f) state property β will hold for a certain time interval of length h. Sometimes, when e=f=g=h=1, a simpler format will be used: $\alpha \twoheadrightarrow \beta$.

3. MONITORING COMPONENT

Within the ambient agent model, a monitoring component has been designed as a specialized Agent Specific Task component. This component receives a stream of information over time, obtained by observation of a world component via the World Interaction Management and Maintenance of World Information components or by communication from other agents, including humans via the Agent Interaction Management and Maintenance of Agent Information components. Typical sources of information are world parts equipped with sensor systems or sensoring agents that interact with such world parts. The component focuses on some properties of the incoming information stream that are to be monitored (*monitoring foci*), e.g., concerning the value of a variable, or a temporal pattern to be detected in the stream. As output it generates information that a certain monitoring focus holds.

A monitor focus can be a state property or a dynamic property. An example of a simple type of state property to be used as a monitor focus is a state property that expresses that the value of a certain variable X is between two bounds LB and UB:

∃V [has_value(X, V) ∧ LB ≤ V ∧ V ≤ UB].

In prefix notation, this can be expressed as follows: exists(V, and(has_value(X, V), and(LB ≤ V, V ≤ UB))). It is possible to obtain abstraction by using (meaningful) names of properties. For example, stable_within(X, LB, UB) can be used as an abstract name for the example property expressed above by specifying:

has_expression(stable_within(X, LB, UB),
exists(V, and(has_value(X, V), and(LB ≤ V, V ≤ UB))))

The fact that a property stable_within(X, LB, UB) is a monitor focus is expressed by: monitor_focus(stable_within(X, LB, UB)). An example of a monitor property is:

∀t [t1≤t ∧ t≤t2 ∧ at(has_value(X, V1), t1) →
∃t', V2 t≤ t' ≤ t+D ∧ V2 ≠V1 ∧ at(has_value(X, V2), t')]

This property expresses that between t1 and t2 the value of variable X is changing all the time, which can be considered as a type of instability of that variable. This dynamic property is expressed in prefix notation as:

forall(t, implies(and(t1≤t, and(t≤t2, at(has_value(X, V1), t))),
exists(t', exists(V2, and(t≤ t', and(t' ≤ t+D, and(V2 ≠V1, at(has_value(X, V2), t'))))

This expression can be named, for example, by

instable_within_duration(X, D).

Within the monitoring component two specific subcomponents are used: Monitoring Foci Determination, and Monitoring Foci Verification.

Monitoring Foci Determination. In this component the monitoring foci are determined and maintained: properties that are the focus of the monitoring task. The overall monitoring foci are obtained from the representation relations derived from the cognitive model representation (see next section). However, to support the monitoring process, it is useful to decompose an overall monitoring focus into more refined foci on particular interaction and world events: its constituents are determined (the subformulas) in a top-down manner, following the nested structure. This decomposition process is specified in the following manner:

monitor_focus(F) ↠ in_focus(F)

in_focus(E) ∧ is_composed_of (E, C, E1, E2)
↠ in_focus(E1) ∧ in_focus(E2)

Here is_composed_of(E, C, E1, E2) indicates that E is an expression obtained from subexpressions E1 and E2 by a logical operator C (i.e., and, or, implies, not, forall, exists). The decomposition process proceeds up to the level of atoms. At each decomposition step subexpressions representing events (i.e., that belong to sort EVENT that comprises names of state properties corresponding to all possible interaction and world events) are added to the list of foci that are used for monitoring. This list is provided to Monitoring Foci Verification component, when the decomposition process is finished.

Monitoring Foci Verification. The process to verify whether a monitoring focus holds, makes use of time-labelled information that is maintained. This is based on information obtained by the World Interaction Management and Agent Interaction Management components of the agent A and are created within these components as follows:

(a) information about the world:

new_world_info(I, S) ∧ current_time(T) ↠
new_world_info(holds_at(I, T), S)

(b) information about another agent:

new_agent_info(I, S) ∧ current_time(T) ↠
new_world_info(holds_at(I, T), S)

Every time when new information about an agent or the world is found, properties from the monitoring foci expressed as TTL formulae, in which this information occurs, are verified automatically on execution histories (or traces) by the TTL Checker tool [8]. In the following the verification algorithm of this tool, is described briefly (for more details see [8]).

The verification algorithm is a backtracking algorithm that systematically considers all possible instantiations of variables in the TTL formula under verification. However, not for all quantified variables in the formula the same backtracking procedure is used. Backtracking over variables occurring in at-formulae is replaced by backtracking over values occurring in the corresponding at-atoms in traces under consideration. Since there are a finite number of such state atoms in the traces, iterating over them often will be more efficient than iterating over the whole range of the variables occurring in the at-atoms. As time plays an important role in TTL-formulae, special attention is given to continuous and discrete time range variables. Because of the finite variability property of traces (i.e., only a finite number of state changes occur between any two time points), it is possible to partition the time range into a minimum set of intervals within which all atoms occurring in the property are constant in all traces. Quantification over continuous or discrete time variables is replaced by quantification over this finite set of time intervals.

The complexity of the algorithm has an upper bound in the order of the product of the sizes of the ranges of all quantified variables. However, if a variable occurs in a at-atom, the contribution of that variable is no longer its range size, but the number of times that the at atom pattern occurs (with different instantiations) in trace(s) under consideration. The contribution of an isolated time variable is the number of time intervals into which the traces under consideration are divided.

Eventually, when a monitoring property E has been satisfied that is an indication for a certain type of abnormal behaviour of the human, the Monitoring agent will indeed believe this:

verification(E, pos) ∧
internal(monitoring_agent)|belief(is_indication_for(E, I))
↠ internal(monitoring_agent)|belief(I)

4. REPRESENTATION RELATIONS

The subcomponent Monitoring Foci Determination of the monitoring component described in the previous section is responsible for the identification of representation relations for cognitive states specified in the represented cognitive model for the human. A representation relation for an internal state property p relates the occurrence of p to a specification Φ that comprises a set of state properties and temporal (or causal) relations between them. In such a case it is said that p *represents* Φ, or Φ describes *representational content* of p. This section presents an automated approach to identify representation relations for cognitive states from a cognitive model representation.

Representation relations for a property p may be defined both backward and forward in time. In the backward case, the representational content is specified by a history (i.e., a specification that comprises temporal (or causal) relations on past states) that relates to the creation of the cognitive state in which p holds. In the forward case, the representational content describes possible (conditional) future states, temporally (or causally) related to the cognitive state in which p holds. In the literature on Philosophy of Mind different approaches to defining representation relations have been put forward; for example, see [13], [4]. For example, according to the classical causal/correlation approach [13], the representational content of an internal state property is given by a one-to-one mapping to an external state property. The application of this approach is limited to simple types of behaviour (e.g., purely reactive behaviour). In cases when an internal property represents a more complex temporal combination of state properties, other approaches have to be used. For example, the temporal-interactivist approach (cf., [4], [12]) allows defining representation relations by referring to multiple (partially) temporally ordered interaction state properties; i.e., input (sensor) and output (effector) state properties over time.

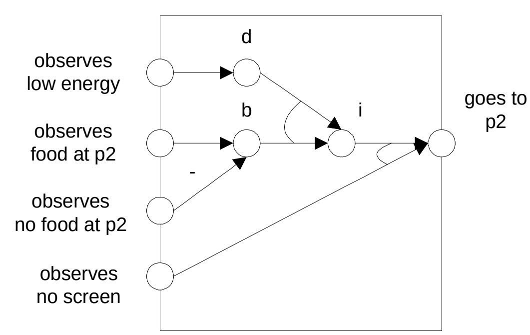

Figure 1. Belief-Desire-Intention (BDI) model
for motivation-based behaviour

An application for the temporal-interactivist approach is demonstrated in the context of the following example of the animal behaviour. Initially, the animal observes that it has low energy (e.g., being hungry). The animal is placed in front of a transparent screen, behind which a piece of food is put afterwards. The animal is able to observe the position of the food and of the screen, after which a cup is place over the food. After some time the screen is raised and the animal chooses to go to the position at which food is present (but invisible). The graphical representation of the cognitive (Belief-Desire-Intention, BDI; see [15]) model that produces such behaviour is given in Figure 1. Here d is the desire to have food, b is the belief that food is present at some position p2, and i is the intention to go to that position.

This paper adopts a rather general specification format for (internal) cognitive models that comprises past-present statements. A *past-present statement* (abbreviated as a *pp-statement*) is a statement φ of the form B ⇔ H, where the formula H, called the *head* and denoted by head(φ), is a statement of the form at(p, t) for some time point t and state property p, and B, called the *body* and denoted by body(φ), is a past statement for t. A *past statement* for a time point t over state ontology Ont is a temporal statement in the reified temporal predicate logic, such that each time variable s different from t is restricted to the time interval before t: for every time quantifier for a time variable s a restriction of the form t > s is required within the statement. Sometimes B is called the *definition* of H.

The cognitive model from the example is formalised by the following properties in past-present format:

IPA1: *Desire d generation*
At any point in time the (persistent) internal state property d holds *iff* at some time point in the past the agent observed its low energy. Formally:

∃t2 [t1 > t2 & at(observed(own_low_energy), t2)] ⇔ at(d, t1)

IPA2: *Intention i generation:*
At any point in time the (persistent) internal state property i holds *iff* at some time point in the past the internal state property d was true, and the internal state property b was true. Formally:

∃t6 t5 >t6 & at(d, t6) & at(b, t6) ⇔ at(i, t5)

IPA3: *Action goto p2 generation:*
At any point in time the agent goes to p2 *iff* at some time point in the past the internal state property i was true and the agent observed the absence of the screen. Formally:

∃t8 t7>t8 & at(observed(no_screen), t8) & at(i,t8)

⇔ at(performing_action(goto_p2),t7)

IPA4: *Belief b generation:*

At any point in time internal state property b holds *iff* at some time point in the past the agent observed that food is present at position p2, and since then did not observe the absence of food. Formally:

∃t10 t9 >t10 & at(observed(food_p2), t10) & ∀t11 t11>t10 & t11<t9 not(at(observed(no_food_p2), t11)) ⇔ at(b1,t9)

Furthermore, a cognitive specification is assumed to be stratified [3], which means that there is a partition of the specification $\Pi = \Pi_1 \cup \ldots \cup \Pi_n$ into disjoint subsets such that the following condition holds: for $i > 1$: if a subformula at(φ, t) occurs in a body of a statement in Π_i, then it has a definition within $\cup_{j<i} \Pi_j$.

The past-present specification format for cognitive models enables that for every internal state property a representation relation can be identified by computing some form of transitive closure. The rough idea is as follows. Suppose for a certain cognitive state property the past-present specification B ⇔ at(p, t) is available (for example IPA2 above). Moreover, suppose that in B only two atoms of the form at(p1, t1) and at(p2, t2) occur, whereas as part of the cognitive model also specifications B1 ⇔ at(p1, t1) and B2 ⇔ at(p2, t2) are available (e.g., variants of IPA1 and IPA4 above). Then, within B the atoms can be replaced (by substitution) by the formula B1 and B2. This results in a

B[B1/at(p1, t1), B2/at(p2, t2)] ⇔ at(p, t)

which again is a past-present specification. Here for any formula C the expression C[x/y] denotes the formula C transformed by substituting x for y.

To automate the process of representation relation identification based on this idea, the following algorithm has been developed:

Algorithm: GENERATE-REPRESENTATION-RELATION

Input: Cognitive specification X; cognitive state specified by at(s, t), for which the representation relation is to be identified, interaction ontology (to specify agent interaction states) InteractOnt, internal ontology (to specify cognitive states) InternalOnt
Output: Representation relation for at(s, t)

1 Stratify X:

1.1 Define the set of formulae of the first stratum (h=1) as
{φ_i: at(a_i, t) ↔ ψi_p(at_1,…, at_m) ∈ X | ∀k m ≥k ≥1 at_k is expressed using InputOnt};
proceed with h=2.

1.2 The set of formulae for stratum h is identified as
{φ_i: at(a_i, t) ↔ ψi_p(at_1,…, at_m) ∈ X | ∀k m ≥k ≥1 ∃l l < h ∃ψ ∈ STRATUM(X, l) AND head(ψ) = at_k AND ∃j m ≥j ≥1 ∃ξ ∈ STRATUM(X, h-1) AND head(ξ)=at_j };
proceed with h=h+1.

1.3 Until a formula of X exists not allocated to a stratum, perform 1.2.

2 Create the stratified specification X' by selecting from X only the formulae of the strata with the number i < k, where k is the number of the stratum, in which at(s, t) is defined. Add the definition of at(s, t) from X to X'.

3 Replace each formula of the highest stratum n of X' φ_i: at(a_i, t) ↔ ψi_p(at_1,…, at_m) by φ_l δ with renaming of temporal variables if required, where δ = {at_k\ body(φ_k) such that φ_k ∈ X' and head(φ_k)=at_k}. Further, remove all formulae { φ ∈ STRATUM(X', n-1) | ∃ψ ∈ STRATUM(X', n) AND head(φ) is a subformula of the body(ψ)})

4 Append the formulae of the stratum n to the stratum n-1, which now becomes the highest stratum (i.e, n=n-1).

5 Until n>1, perform steps 3 and 4. The obtained specification with one stratum (n=1) is the representation relation specification for at(s, t)

In Step 3 subformulae of each formula of the highest stratum n of X' are replaced by their definitions, provided in lower strata. Then, the formulae of n-1 stratum used for the replacement are eliminated from X'. As result of such a replacement and elimination, X' contains n-1 strata (Step 4). Steps 3 and 4 are performed until X' contains one stratum only. In this case X' consists of a formula φ defining the representational content for at(s, t), i.e., head(φ) is at(s, t) and body(φ) is a formula expressed over interaction states and (temporal) relations between them. In the following it is shown how this algorithm is applied for identifying the representational content for state i from the

example. By performing Step 1 the specification of the cognitive model given above is automatically stratified as follows: stratum 1: {IPA1, IPA4}; stratum 2: {IPA2}; stratum 3: {IPA3}.
By Step 2 the property IPA3 is eliminated as unnecessary for determining the representational content of i.
Further, in Step 3 we proceed with the property IPA2 of the highest stratum that defines the internal state i.

∃t6 t5 >t6 & at(d, t6) & at(b1, t6) ⇔ at(i, t5)

By Step 3 both d and b1 state properties are replaced by their definitions with renaming of temporal variables in the stratum 1. The property IP2 is replaced by the following formula:

∃t6 t5 >t6 & ∃t2 t6 > t2 & at(observed(own_low_energy), t2) &
∃t10 t6 >t10 & at(observed(food_p2), t10) & ∀t11 t11>t10 & t11<t6
& not(at(observed(no_food_p2), t11)) ⇔ at(i, t5)

Further, both formulae IPA1 and IPA4 are removed and the property resulted from the replacement is added to the stratum 1, which becomes the only stratum in the specification. The obtained formula is the representational content for the state i that occurs at any time point t5.

The algorithm has been implemented in Java. Worst case time and representation complexity of the algorithm are satisfactory as will be briefly discussed. The worst case time complexity of the algorithm is estimated as follows. The worst case time complexity for step 1 is $O(|X|^2/2)$. Time complexity of step 2 is $O(|X|)$. The worst case time complexity for steps 3-5 is calculated as:

O(|STRATUM(X', n)|· | STRATUM(X', n-1)|)+ O(|STRATUM(X', n)|· |STRATUM(X', n-2)|) + … + O(|STRATUM(X', n)|· |STRATUM(X', 1)|) = O(|STRATUM(X', n)|·|X'|).

Thus, the overall time complexity of the algorithm for the worst case is $O(|X|^2)$.

5. EXAMPLES

This section illustrates the approach for the automated identification of representation relations content proposed in Section 4 by two examples of cognitive models.

In the first example a model based on the theory of consciousness by Antonio Damasio [11] is considered. In particular, the notions of 'emotion', 'feeling', and 'core consciousness' or 'feeling a feeling' are addressed. Damasio [11] describes an emotion as neural object (or internal emotional state) as an (unconscious) neural reaction to a certain stimulus, realised by a complex ensemble of neural activations in the brain. As the neural activations involved often are preparations for (body) actions, as a consequence of an internal emotional state, the body will be modified into an externally observable state. Next, a feeling is described as the (still unconscious) sensing of this body state. Finally, core consciousness or feeling a feeling is what emerges when the organism detects that its representation of its own body state (the proto-self) has been changed by the occurrence of the stimulus: it becomes (consciously) aware of the feeling. In Figure 2 a cognitive model for this process is depicted. Here s0 is an internal representation of the situation that no stimulus is sensed, and no changed body state, s1 is an internal representation of the sensed stimulus without a sensed changed body state yet, and s2 is an indication for both sensed stimulus and changed body state (which is the core consciousness state).

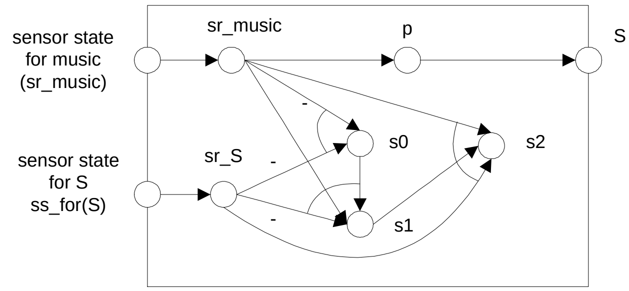

Figure 2. Cognitive model based on the theory of core consciousness by Damasio [11]

The cognitive model for this example comprises the following properties expressed in past-present format:

LP1: *Generation of the sensory representation for music*
At any point in time the sensory representation for music occurs *iff* at some time point in the past the sensor state for music occurred. Formally:
∃t2 t1 > t2 & at(sr_music, t2) ⇔ at(sr_music, t1)

LP2: *Generation of the preparation*
At any point in time the preparation p occurs *iff* at some time point in the past the sensory representation for music occurred. Formally:

∃t4 t3 > t4 & at(sr_music, t4) ⇔ at(p, t3)

LP3: *Generation of the body state*
At any point in time the body state S occurs *iff* at some time point in the past the preparation p occurred. Formally:

∃t6 t5 > t6 & at(p, t6) ⇔ at(S, t5)

LP4: *Generation of the sensor state*
At any point in time the sensor state for S occurs *iff* at some time point in the past the body state S occurred. Formally:

∃t8 t7 > t8 & at(S, t8) ⇔ at(ss_for(S), t7)

LP5: *Generation of the sensory representation for S*
At any point in time the sensory representation for S occurs *iff* at some time point in the past the sensor state vector for S occurred

Formally: ∃t10 t9 > t10 & at(ss_for(S), t10) ⇔ at(sr_S, t9)

LP6: *Generation of s0*
At any point in time s0 occurs *iff* at some time point in the past no sensory representation for music and no sensory representation for S occurred. Formally:

∃t12 t11 > t12 & not(at(sr_music, t12)) & not(at(sr_S, t12)) ⇔ at(s0, t11)

LP7: *Generation of s1*
At any point in time s1 occurs *iff*
at some time point in the past the sensory representation for music and no sensory representation for S and s0 occurred. Formally:

∃t14 t13 > t14 & at(sr_music, t14) & at(s0, t14) & not(at(sr_S, t14)) ⇔ at(s1, t13)

LP8: *Generation of s2*
At any point in time s2 occurs *iff*
at some time point in the past the sensory representation for music and the sensory representation for S and s1 occurred. Formally:
∃t16 t15 > t16 & at(sr_music, t16) & at(s1, t16) & at(sr_S, t16) ⇔ at(s2, t15)

The generated representation relation for state s2 is:

exists(t16) t15>t16 exists(t2) t16>t2 at(ss_music,t2) & exists(t10) t16>t10 at(ss_for(S),t10) & exists(t14) t16>t14 exists(t2') t14>t2' at(ss_music,t2') & not(exists(t10') t14>t10' at(ss_for(S),t10')) & exists(t12) t14>t12 not(exists(t2'') t12>t2''

& at(ss_music,t2")) & not(exists(t10") t12>t10" at(ss_for(S),t10"))
⇔ at(s2,t15)

The model presented in the second example is used for the simulation described in Section 6. This example considers incremental improving skills by a human with every subsequent performance of an activity or of a task a. The level of skill related to a may be indicated by a value val (i.e., skill(a, val)). Initially, when the human receives a request for a new activity or task, s/he has the initial skill level indicated by init, which may be numerical. For each observed request a sensory representation state is created. Furthermore, for a requested activity a with which the human is not acquainted, a sensory representation state is created indicating that a is new for the human. This is specified by the following properties:

IP1: *Sensory representation of a request*
At any point in time the internal sensory representation property for rq(a) holds *iff* at some time point in the past the human observed that a is requested to be performed. Formally:

∃t2 t1 > t2 & at(observed(rq(a), t2) ⇔ at(sr(rq(a)), t1)

IP2: *Sensory representation of a new activity*
At any point in time the internal sensory representation property for new(a) holds *iff* at some time point in the past the human observed that a is new. Formally:
∃t4 t3 > t4 & at(observed(new(a), t4) ⇔ at(sr(new(a)), t3)

IP3: *Initial value of the skill and its persistency*
At any point in time the internal state property skill(a, init) indicating the initial skill for a holds *iff* at some time point in the past the internal state property sr(new(a)) was true and since then no observation of finished(a, init) occurred indicating that a has been performed by the human with the initial (minimal) skill level. Formally:

∃t6 t5 > t6 & at(sr(new(a), t6) & ∀t7 t7>t6 & t5 > t7 ⇒ not(at(observed(finished(a, init)), t7)) ⇔
at(skill(a, init), t5)

With every following execution of a, after observing the results of the execution, the human gains an increment of the skill level. A (estimated) qualitative/quantitative value of the increment depends on the type of the activity, characteristics of the human and the environment, in which the activity is being executed. Sometimes, a functional dependency of the gained skill related to the previous skill can be determined. The following property describes how the gained skill level can be determined in a generic way given a functional dependency *f* of the gained skill from the previous skill value. Furthermore, it is assumed that the maximal value of skill indicated by max exists for the human, which once gained will persist.

IP4: *Generation of an intermediate skill state and its persistency*
At any point in time the internal state property skill(a, v) indicating the gained skill level of the human with a with the value specified by the numerical variable v holds, *iff* at some time point in the past the internal state property skill(a, v1) indicating the previous human's skill with a with f(v1)=v was true and the human observed finished(a, v1), i.e., that his/her execution of a with the skill level specified by variable v1 finished (with some results), and after this time point no observation of finished(a, v) occurred. Formally:

∃t8 t9 > t8 & v>init & v< max &
∃v1 & f(v1)=v & at(skill(a, v1), t8) & at(observed(finished(a, v1)), t8) & ∀t10 t10>t8 & t9 > t10 not(at(observed(finished(a, v)), t10)) ⇔
at(skill(a, v), t9)

When the function f is known, the property IP4 can be instantiated automatically in a number of more specific rules with bound variables v and v1, starting with v1 = init. This step is required for automated generation of the representational content for a state indicating skill. For illustration purposes one intermediate skill state is considered with the skill level specified by constant val (see the property IP4i below). However, in general the property IP4 may be applied for generating any number of intermediate skill states.

IP4i: *Generation of the intermediate skill state and its persistency*

∃t8 t9 > t8 & at(skill(a, init), t8) & at(observed(finished(a, init)), t8) & ∀t10 t10>t8 & t9 > t10 not(at(observed(finished(a, val)), t10)) ⇔
at(skill(a, val), t9)

IP5: *Generation of the maximal skill state and its persistency*
At any point in time the internal state property skill(a, max) holds *iff* at some time point in the past the internal state property skill(a, v1) indicating the human's skill with a with f(v1)=max was true and the human observed finished(a, v1), i.e. that his/her execution of a with the skill level specified by v1 finished. Formally:

∃v1 f(v1)=max ∃t12 t11 > t12 & at(skill(a, v1), t12) & at(observed(finished(a, v1)), t12) ⇔ at(skill(a, max), t11)

Considering one intermediate skill state used in the example, this property is transformed automatically into:

∃t12 t11 > t12 & at(skill(a, val), t12) &
at(observed(finished(a, val)), t12) ⇔ at(skill(a, max), t11)

Before performing an activity a preparation state is generated. The generation of preparation states and subsequent execution states is described by the following properties.

IP6: *Preparation state generation*
At any time point the internal preparation state property preparation_for(a, v) for the execution of a with the skill level specified by the variable v holds *iff* at some time point in the past the internal state property sr(rq(a)) was true and the internal state property skill(a, v) was true. Formally:

∃t14 t13 > t14 & at(sr(rq(a)), t14) & at(skill(a, v), t14) ⇔
at(preparation_for(a, v), t13)

IP7: *Action state generation*
At any time point the action state specified by performed(a, v) holds *iff* at some time point in the past the internal preparation state property preparation_for(a, v) was true. Formally:

∃t16 t15 > t16 & at(preparation_for(a, v), t16) ⇔
at(performed(a, v), t15)

The properties IP6 and IP7 can be easily instantiated for any skill value of v. The graphical representation of the model considered is given in Figure 3. Using the described cognitive model, the representational content of the internal state skill(a,max) is automatically generated as:

∃t12 t11>t12 & ∃t8 t12>t8 & ∃t6 t8>t6 & ∃t4 t6>t4 &
at(observed(new(a),t4) & ∀t7 t7>t6 & t8>t7 &
not(at(observed(finished(a,init)),t7)) &
at(observed(finished(a,init)),t8) & ∀t10 t10>t8 & t12>t10 &
not(at(observed(finished(a,val)),t10)) &
at(observed(finished(a,val)),t12)
⇔ at(skill(a, max), t11)

The next section illustrates how the generated representation content representation is used for monitoring and verification by the monitoring component in the context of a simulation case.

6. SIMULATION RESULTS

This section illustrates the functioning of an ambient agent in the context the last example from the previous Section 5, in which skill learning of a human in certain activities is considered. The considered simulation case has a special focus on the monitoring component of the ambient agent. To obtain necessary information (e.g., a cognitive model, beliefs) the monitoring component interacts with other components of the agent.

In particular, the cognitive model describing incremental gaining of skill stored as sets of beliefs in the Maintenance of Agent Information component is transferred to the Monitoring Foci Determination component for further processing. Beliefs about the world observation stored in the World Interaction Management and about communication with other agents stored in the Maintenance of Agent Information component are transmitted to Monitoring Foci Verification component. During the internal information transmission one-to-one mapping of beliefs occurs.

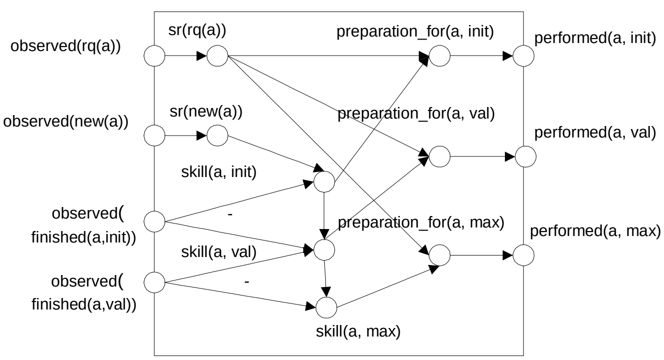

Figure 3. Cognitive Model describing incremental gaining of skill

In the considered simulation case the Monitoring component aims at identifying the time point, at which the internal state of the human is generated indicating that the maximal skill with a is gained (i.e., time point t such that at(skill(a, max), t11) holds). To this end, the monitoring foci determination subcomponent of Monitoring Component generates the representational content for this state as shown in the end of Section 5 using the cognitive model for gaining skill (see Section 5). Then, to enable verification of monitoring focuses on beliefs stored in the agent, each subformula of the representational content of form at(I, t), where I is a state property and t is a temporal variable is replaced by belief(holds_at(I, t), pos). Further, the monitoring foci determination subcomponent extracts automatically atomic monitoring foci (i.e., events to be observed) using the procedure described in Section 3, which are provided to the monitoring foci verification subcomponent. For the considered case study the extracted atomic monitoring foci are:

observed(new(a))),
observed(finished(a,init))
observed(finished(a,val)).

Also, the whole property in focus is provided to the monitoring foci verification subcomponent. To enable automated verification of the property in focus by the TTL Checker tool described in Section 3, the provided property is translated automatically into the input format of this tool by the monitoring foci verification subcomponent. In particular, the property in focus from the case study is automatically translated into:

denotes(repr_content, exists([t12:interval, t8:interval, t6:interval, t4:interval, t21:interval, t22:interval, t23:interval, t24:interval, t25:interval], and(t12>t8, t8>t6, t6>t4, holds(state(trace1, time(t21:interval)), belief(holds_at(observed(new(a)), t4), pos), true), holds(state(trace1, time(t22:interval)), belief(holds_at(observed(finished(a, init)), t8), pos), true), holds(state(trace1, time(t23:interval)), belief(holds_at(observed(finished(a, val)), t23), pos), true), forall([t7:interval], and(t7>t6, t8>t7, holds(state(trace1, time(t24:interval)), belief(holds_at(observed(finished(a, init)), t7), neg), true))), forall([t10:interval], and(t10>t8, t12>t10, holds(state(trace1, time(t25:interval)), belief(holds_at(observed(finished(a, val)), t10), neg), true)))))).

The monitoring foci verification subcomponent constantly monitors the belief base of the agent. As soon as a belief about an event that is in the atomic monitoring foci occurs, the subcomponent initiates automated verification of the complete property in focus on the history of the events in focus occurred so far.

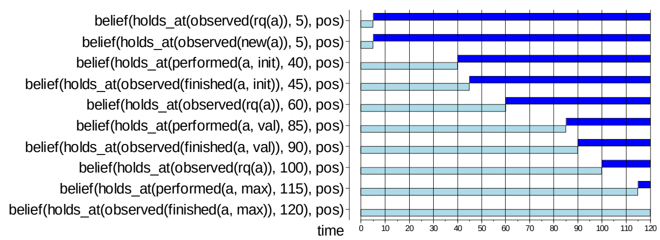

Figure 4. A history (trace) of events in focus used for verification

For this case such a history (or a trace) was created using the LEADSTO simulation tool [7]. A part of the used trace is shown in Figure 4. When the complete property in focus that corresponds to the representation content is established to hold, the belief of the agent is created that the human has gained the maximal skill with a and further actions may be undertaken.

7. DISCUSSION

In this paper an ambient agent model was presented for automated mindreading based on monitoring of a human's interaction with the environment. Within this agent model, monitoring foci are determined from explicitly represented cognitive models by automatically deriving representation relations for cognitive states in the form of temporal predicate logical specifications. From these temporal expressions the events are derived that are to be monitored, and from the monitoring information on these events the representation expressions are verified automatically.

The method put forward is very general in the sense that for any cognitive model it can be applied, under the condition that it can be stratified. The latter condition excludes models in which loops occur that potentially can be processed an unbounded number of times. In future work it will be investigated in how far also such loops can be handled under certain assumptions, for examples by adding a bound on the number of times it is actually processed.

The method on verification was based on the TTL environment. In [6] also a verification approach was developed originating in a similar idea, but worked out in the context of the LEADSTO language. Another, even bigger difference is that in the latter paper an monitoring focus was assumed, while in the current paper a method is presented to automatically generate it from any cognitive model.

8. REFERENCES

[1] Aarts, E., Collier, R., van Loenen, E., Ruyter, B. de (eds.) 2003. Ambient Intelligence. Proc. of the First European Symposium, EUSAI 2003. Lecture Notes in Computer Science, vol. 2875. Springer Verlag, pp. 1-432.

[2] Aarts, E., Harwig, R. and Schuurmans, M. 2001. Ambient Intelligence. In: P. Denning (ed.), The Invisible Future, McGraw Hill, New York, pp. 235-250.

[3] Apt, K.R., Blair, H.A., Walker, A. 1988. Towards a Theory of Declarative Knowledge. Foundations of Deductive Databases and Logic Programming, 89-148

[4] Bickhard, M.H. 1993. Representational Content in Humans and Machines. Journal of Experimental and Theoretical Artificial Intelligence, 5, 285-333.

[5] Bosse, T., Hoogendoorn, M., Klein, M., and Treur, J. 2008. An Agent-Based Generic Model for Human-Like Ambience. In: Proceedings of the First International Workshop on Model Driven Software Engineering for Ambient Intelligence Applications. Published in: M. Mühlhäuser, A. Ferscha, and E. Aitenbichler (eds.), Constructing Ambient Intelligence: AmI-07 Workshops Proceedings. Communications in Computer and Information Science (CCIS), vol. 11, Springer Verlag, 93-103.

[6] Bosse, T., Hoogendoorn, M., Klein, M., and Treur, J. 2008. A Component-Based Ambient Agent Model for Assessment of Driving Behaviour. In: Proceedings of the 5th International Conference on Ubiquitous Intelligence and Computing. Lecture Notes in Computer Science, Springer Verlag, to appear.

[7] Bosse, T., Jonker, C. M., van der Meij, L., and Treur, J. 2007. A Language and Environment for Analysis of Dynamics by Simulation. International Journal of Artificial Intelligence Tools, vol. 16, 435-464.

[8] Bosse, T., Jonker, C.M., Meij, L. van der, Sharpanskykh, A, and Treur, J. 2006. Specification and Verification of Dynamics in Cognitive Agent Models. In: Nishida, T. et al. (eds.), Proceedings of the Sixth International Conference on Intelligent Agent Technology, IAT'06. IEEE Computer Society Press, 247-254.

[9] Bosse, T., Jonker, C.M., Meij, L. van der, and Treur, J. 2007. A Language and Environment for Analysis of Dynamics by Simulation. International Journal of Artificial Intelligence Tools, vol. 16, 435-464.

[10] Brazier, F.M.T., Jonker, C.M., and Treur, J. 2000. Compositional Design and Reuse of a Generic Agent Model. Applied Artificial Intelligence Journal, vol. 14, 2000, 491-538.

[11] Damasio, A. 2000. The Feeling of What Happens: Body, Emotion and the Making of Consciousness. MIT Press.

[12] Jonker, C.M., and Treur, J. 2003. A Temporal-Interactivist Perspective on the Dynamics of Mental States. Cognitive Systems Research Journal, vol. 4, 137-155.

[13] Kim, J. 1996. Philosophy of Mind. Westview Press.

[14] Manzano, M. 1996. Extensions of First Order Logic, Cambridge University Press.

[15] Rao, A. and Georgeff, M.P. 1991. Modeling rational agents within a bdi-architecture. In Proceedings of 2th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR:91), 473-484, Cambridge, MA.

[16] Reiter, R. 2001. Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press.

[17] Riva, G., Vatalaro, F., Davide, F., Alcañiz, M. (eds.). 2005. Ambient Intelligence. IOS Press.

[18] Sharpanskykh, A. and Treur, J. 2006. Verifying Interlevel Relations within Multi-Agent Systems. In Proceedings of the 17th European Conference on Artificial Intelligence, IOS Press, 247-254.